CHAPTER 27

Adding Styles to Your Web Pages

In This Chapter

Using an external style sheet

Using a STYLE element within the HEAD

Adding inline styles

Using a standardized style sheet

Choosing the right approach

Now that you know why you should use a style sheet and how to write a style sheet, you need to know how to include style sheets in your page. This document takes you through the three options you have for adding styles to your page. With the cascading approach, you can actually use all three at the same time, but you probably don't want to do this, as you learn in this chapter.

Using an External Style Sheet

The preferred way to include styles in your page is to use an external style sheet. This is the only way to take advantage of all the benefits you have read about relating to using style sheets. If you use an external style sheet, then all your style information for all your pages is stored in one central place. If you want to give your site a face-lift, it's as easy as changing the external style sheet.

To use an external style sheet, simply create your style sheet and save it with a .css extension. Then link to it from the HEAD of your page using the LINK element. The following list is not the complete specification for the LINK element. The LINK element is used for other unrelated functions, which only confuse this use.

For the complete specification of the LINK element, refer to Chapter 12.

Link <LINK>

Start Tag: Required

Content: Empty

End Tag: Forbidden

Attributes: id, class, lang, dir, title, style, events

href: URL of style sheet

hreflang: language of document specified in href attribute

type: MIME type of document specified in href attribute, usually "text/css"

Here is an example of the LINK element used to link to an external style sheet called may2000.css that exists in the styles subdirectory:

```
<HEAD>
<LINK href="styles/may2000.css" type="text/css">
</HEAD>
```

The LINK element, whether it is used for linking to an external style sheet or for some other purpose, must be located in the HEAD element.

Using a STYLE Element within the HEAD

If you are unhappy with any pre-existing style sheet or you don't want to store your styles in a separate style sheet for whatever reason, then you can store your complete style sheet in the HEAD element directly. How does this work? You put the rules (the statements that tell the browser how to render each element) directly into a STYLE element in the HEAD. As you can imagine, this could make your HEAD quite long.

The STYLE element has the following specification:

Style <STYLE>

Start Tag: Required

Content: Rules

End Tag: Required

Attributes: lang, dir

type: MIME type, usually "text/css"

media: types of medium for which this style sheet is relevant; defaults to "screen"; must be a keyword or comma-delimited list of media keywords from list (see following valid list)

title: title of style information

The valid `media` types are as follows:

- **screen.** For computer screens.
- **tty.** For fixed-width terminal displays.
- **tv.** For televisions.
- **projection.** For projectors.
- **handheld.** For handheld devices.
- **print.** For print.
- **braille.** For braille-tactile feedback devices.
- **aural.** For speech synthesizers.
- **all.** For all devices.

The `STYLE` element is used as follows:

```
<HEAD>
<STYLE>
  BODY {
       color: black;
       background: white;
       padding-left: 1in;
       padding-right: 1in;
  }
  H1 {
       font: bold 24pt/30pt black;
  }
  P {
       text-indent: .5in;
  }
  A:link {
       color: blue;
  }
  A:visited {
       color: red;
  }
  A:active {
       color: yellow;
  }
  BLOCKQUOTE {
       margin-left: 1in;
       margin-right: 1in;
  }
</STYLE>
</HEAD>
```

Adding Inline Styles

Reasons may exist why you would want to add style information directly at the element level. You may have noticed that nearly every element introduced in this book has had a `style` attribute. This attribute is used to define element-specific style information. The `style` attribute is used as follows:

```
<P style="text-indent: 1in; color: blue;">This paragraph has a
one inch indentation--twice the normal paragraph indentation--
on the first line and is rendered with blue text.</P>
```

Notice you don't need a selector (`P`) in the `style` attribute because you are only defining a rule for this instance of the element. You do need double quotes around the rules, as with any attribute value, and each descriptor does need to be followed by a semicolon.

The use of the `style` attribute is preferred to the use of the `FONT` element, which is deprecated in HTML 4, but the `style` attribute is still not the ideal. Whenever possible, you should try to determine what formatting you need before you get to the element that needs special formatting and define a class for that element in your style sheet. The previous example could have been written instead as:

```
<P class="deep-blue">This paragraph has a one inch indentation--
twice the normal paragraph indentation--on the first line and is
rendered with blue text.</P>
```

A rule would have to be in your style sheet as well:

```
P.deep-blue {
     text-indent: 1in;
     color: blue;
}
```

Either example renders the same in the browser. Either one enables your page to be rendered easily in speech-synthesizing browsers. Both are vastly preferable to the HTML 3.2 method of using a clear GIF to indent the paragraph and then using the `FONT` element to change the text color to blue.

Using a Standardized Style Sheet

If after all these style sheets examples, you still don't feel comfortable writing your own style sheet, don't despair. Many standard style sheets are available. Most people get started writing HTML by copying pages with features they like and modifying the code to meet their needs. Most people who write Perl use this

technique; most people who write JavaScript got started this way. Finding a style sheet you like and modifying it to meet your needs is no disgrace.

To help make it easier, the CD-ROM in the back of this book has several standard style sheets on it. Pick one that comes close to approximating the look you want and customize it to make it your own.

You might find your organization has its own style sheet, which you are expected to use. If this is the case, the powers that control the style sheet might not let you create special classes to accommodate your formatting needs. Fortunately, CSS has a way to help you get around any limitations this might present. Consider the following HTML, which refers to a corporate style sheet. It doesn't even matter what is in the style sheet for your purposes because you are required to use it and you are not allowed to change it.

```
<HTML>
<HEAD>
<LINK href="style/corporate.css" type="text/css">
</HEAD>

<BODY>

<H1>A heading</H1>

<P>This paragraph looks just the way they want it to look.</P>

<P>But I sure wish I could make this one look different.</P>

</BODY>
</HTML>
```

You can see the frustration you would encounter because you can't change the formatting of the second paragraph using the corporate style sheet you are provided. Given what you've learned in this chapter, you know two ways to get around this limitation. The first is to create a `STYLE` element in the `HEAD` element and create a class for the second paragraph.

```
<HTML>
<HEAD>
<LINK href="style/corporate.css" type="text/css">
<STYLE>
  P.different {
       color: blue;
       font-size: 14pt;
  }
</STYLE>
</HEAD>
```

```
<BODY>

<H1>A heading</H1>

<P>This paragraph looks just the way they want it to look.</P>

<P class="different">I'm so pleased that I can make this one
look different.</P>

</BODY>
</HTML>
```

The other way you learned is to put the style information directly into a `style` attribute in the `P` element.

```
<HTML>
<HEAD>
<LINK href="style/corporate.css" type="text/css">
</HEAD>

<BODY>

<H1>A heading</H1>

<P>This paragraph looks just the way they want it to look.</P>

<P style="color: blue; font-size: 14pt;">I'm so pleased that I
can make this one look different.</P>

</BODY>
</HTML>
```

For the previous example, where the customization you want to affect in formatting is minimal and only affects one element, either of the two methods of accomplishing this would be fine. Neither one substantially clutters your HTML. The first one is probably better, but the difference is minuscule. However, there is one other way to take control of the look of your page at a micro level when you are forced to use a style sheet over which you have no control. This is to create another external style sheet and link to it in the `HEAD` following the link to the standard style sheet. If, however, you want to create classes or change formatting for multiple elements, you probably want to move this formatting to your own style sheet, as shown in the following:

```
<HEAD>
<LINK href="style/corporate.css" type="text/css">
<LINK href="style/mine.css" type="text/css">
</HEAD>
<BODY>
<H1>A heading</H1>
```

```
<P>This paragraph looks just the way they want it to look.</P>

<P class="different">I'm so pleased that I can make this one
look different.</P>

</BODY>
</HTML>
```

You can actually include as many links to external style sheets as you need. Does the order matter? You bet! Because of the cascading rules discussed in the last chapter, the last style sheet specified can override all previous style sheets specified. If you are supposed to use a standard style sheet created by someone else, then you want to be careful not to override any of those styles. As with all style definitions, the lowest-level style defined has the highest priority. Conversely, the higher a style is defined, the lower the priority.

Mixing the Approaches: An Example

The following example of combining style definitions should help make the concept of cascading style sheets and the use of classes, IDs, and inline style definitions clear. This is a long example, but it is one to which you can refer later if you have questions about your own implementation of style sheets.

Corporate.css

```
BODY {
     color: black;
     background: white;
     padding-left: 1in;
     padding-right: 1in;
}
H1 {
     font: bold 24pt/30pt black;
}
P {
     text-indent: .5in;
}
A:link {
     color: blue;
}
A:visited {
     color: red;
}
A:active {
     color: yellow;
}
BLOCKQUOTE {
     margin-left: 1in;
```

```
        margin-right: 1in;
    }

mine.css
    SPAN.special {
        color: red;
    }

    #wide {
        letter-spacing: .5em;
    }
    HR {
        width: 3pt;
        color: red;
    }
```

The two external style sheets used in this example are defined previously. They are called `corporate.css` and `mine.css`. The HTML page is defined in the following:

```
<HTML>
<HEAD>
<LINK href="corporate.css" rel="stylesheet" type="text/css">
<LINK href="mine.css" rel="stylesheet" type="text/css">
<STYLE>
  H1#wide {
        line-height: 40pt;
  }
</STYLE>
</HEAD>
<BODY>
<H1 ID="wide">A page with <SPAN class="special">special</SPAN>
formatting</H1>
<P>This paragraph contains <SPAN id="wide">some rather widely-
spaced text.</SPAN></P>
<HR style="color: black">
<HR>
<P>This paragraph is normal, but you can <A href="weird.html">
link</A> to weird stuff from here.</P>
</BODY>
</HTML>
```

In the previous example, many conflicting instructions exist in the form of style information for the browser. How will it render?

The first `H1` has an `ID` of wide. If you look at `mine.css`, you can see an `ID` of wide is defined that affects letter spacing. But this won't be used. Why? Because another `ID` of wide specific to the `H1` element is defined in the `STYLE` element and that `ID` affects line height. Notice the `H1` element also has a `SPAN` element with a `class` of special that affects one word. The special class is defined in `mine.css` and isn't

overridden anywhere further down, so the word `special`, within the `SPAN` element, will be rendered as red.

The first paragraph also uses the `SPAN` element with an `ID` of wide. How will this render? It will use the `ID` wide defined in `mine.css`, because it isn't affected by the `H1` element defined in the `STYLE` element.

The first `HR` element will render in black, because the `style` attribute is defined right there and will override the `HR` definition in `mine.css`. It will render with the `width` provided in `mine.css`, since the new definition doesn't specify the `width`. How about the second `HR` definition? Will it inherit the color of the previous `HR` element? No. It will render just as `mine.css` defines it because no style information overrides the style information in `mine.css`.

Finally, in the last paragraph, which doesn't have any style references, there is a link. How will the link render? Until the link is visited, it will render in blue as defined by `A:link` in `corporate.css`. Once it is visited, it will render in red, and when it is active, it will render in yellow. Nothing special must be done to the HTML to take advantage of the three pseudo-elements related to the `A` element.

From Here

Jump to Chapter 33 and learn about absolute positioning in CSS.

Proceed to Chapter 28 and learn about adding colors and backgrounds to your pages.

Summary

In this chapter you got a crash course in using CSS in your page. You learned how to link to an external style sheet. You learned about using the `STYLE` element and you learned about using inline style definitions. You learned about using a standardized style sheet and how you can get around any limitations related to this. If you persevered, you also learned how cascading rules affect the implementation of styles in your page in a thorough example.

✦ ✦ ✦